Efficient FPGA-Based DSP up to GS/s

FPGA Conference Europe 2022

Dr. Harry Commin | Enclustra GmbH | 05 July 2022

Agenda

- *Enclustra company introduction*
- *Traditional DSP design challenges*
- *Parallel DSP design challenges*
  - *Parallel FIR filter*
  - *Parallel FFT*

Focused on FPGA Technology – Everything FPGA!

- *FPGA hardware*
- *HDL firmware*
- *Embedded software*

>80 employees, based in Zurich.

Branch offices in Germany and China.

- *Sales representatives worldwide.*

XILINX. ALLIANCE PROGRAM CERTIFIED MEMBER — BASE

intel FPGA Design Solutions Network Gold

LEADER Lattice Exclusive Alliance of Design Engineering Resources

Microsemi

MathWorks Partner

Focused on FPGA Technology – Everything FPGA!

- *FPGA hardware*
- *HDL firmware*
- *Embedded software*

>80 employees, based in Zurich.

Branch offices in Germany and China.

- *Sales representatives worldwide.*

Key Target Markets

ROBOTICS

VISION

SECURITY

MEASUREMENT

INDUSTRIAL

MEDICAL

FPGA **Design Services**
Customer-Specific Designs

FPGA **Standard Products**
FPGA/SoC Hardware Modules and
IP Solutions

Well-balanced know-how.

FPGA **Design Services**

Customer-Specific Designs

FPGA **Standard Products**

FPGA/SoC Hardware Modules and

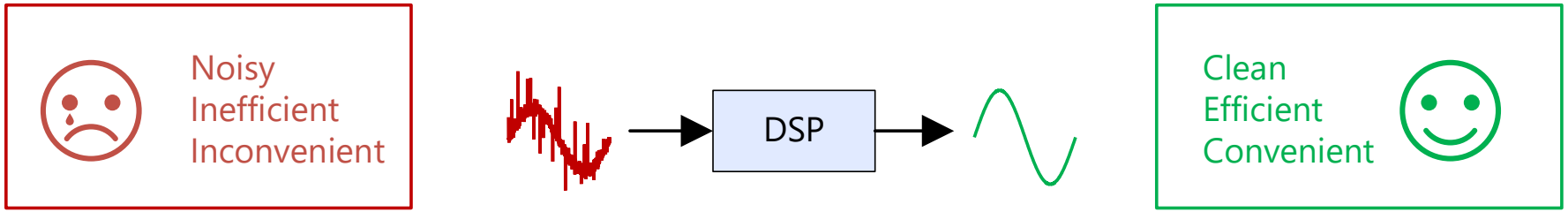
IP Solutions

Well-balanced know-how.

ENCLUSTRA
FPGA SOLUTIONS

Traditional DSP

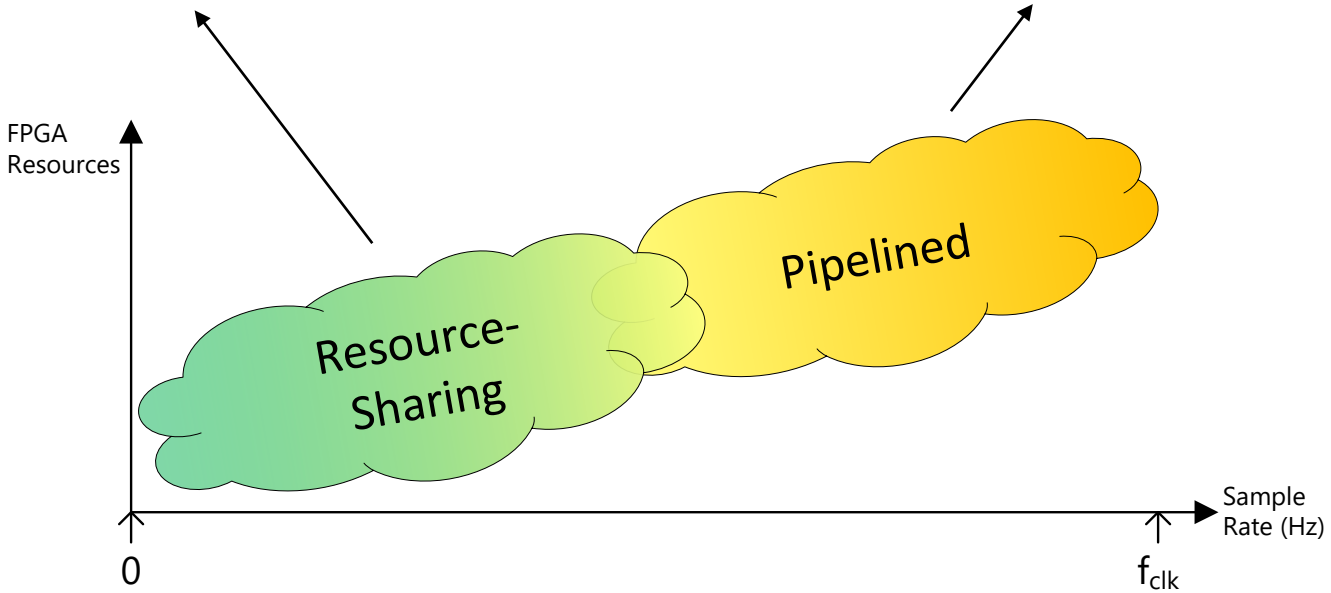
- *What is Digital Signal Processing?*



- *Processing each sample requires processing effort*
- *Higher sample rate ⇒ higher effort rate ⇒ more FPGA resources*

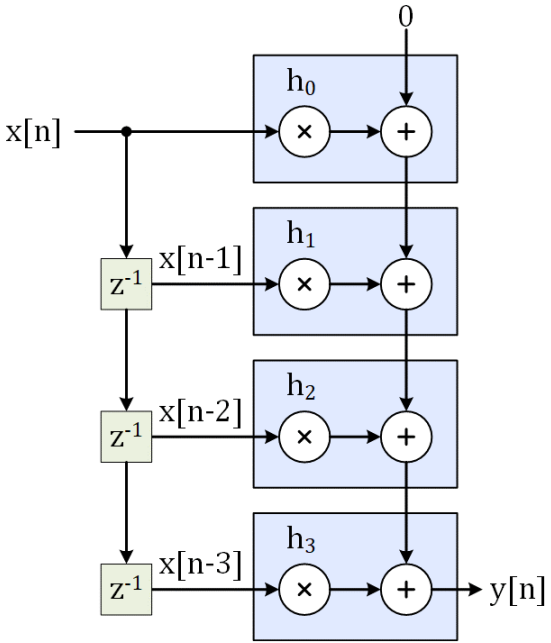
Traditional DSP Tradeoffs

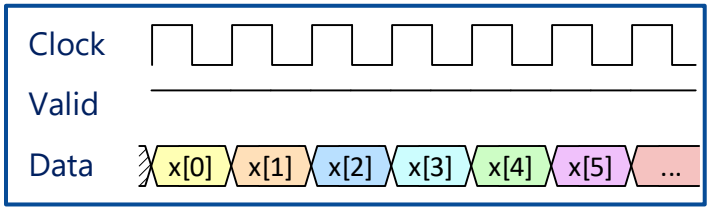
- *Resource-sharing designs*
  - *Lower resource consumption* ☺
  - *Lower processing power* ☹
- *Pipelined designs*
  - *Higher resource consumption* ☹
  - *Higher processing power* ☺



Traditional DSP Example

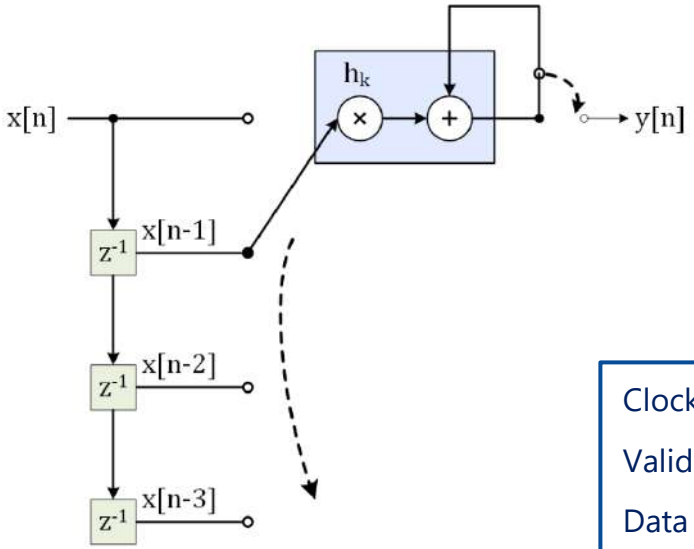
- ***Example***: *Pipelined FIR Filter*
  - *Ready for valid input every clock cycle*

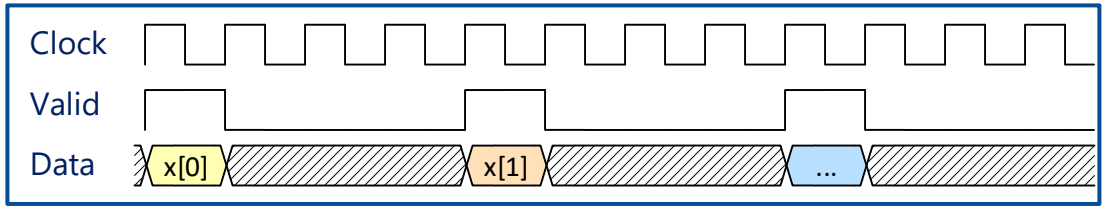




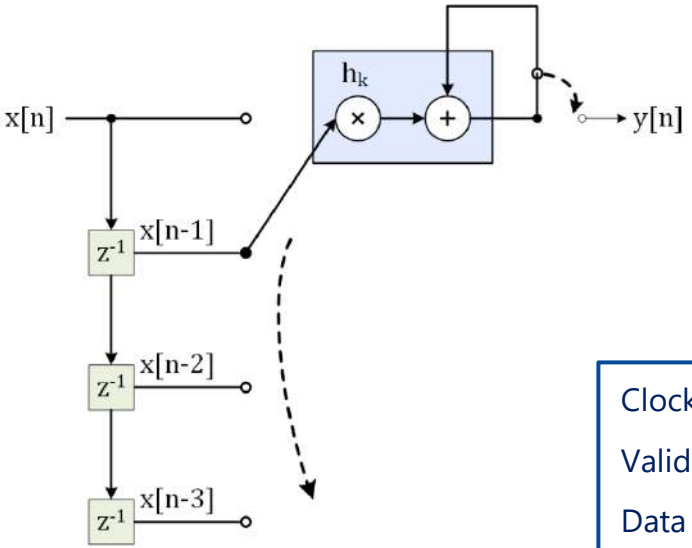
Traditional DSP Example

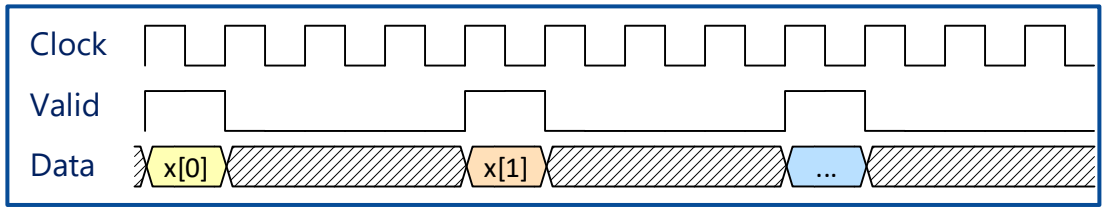
- ***Example**: Resource-sharing FIR Filter*
  - *Ready for valid input every 4 clock cycles*





- ***Example**: Resource-sharing FIR Filter*
  - *Ready for valid input every 4 clock cycles*





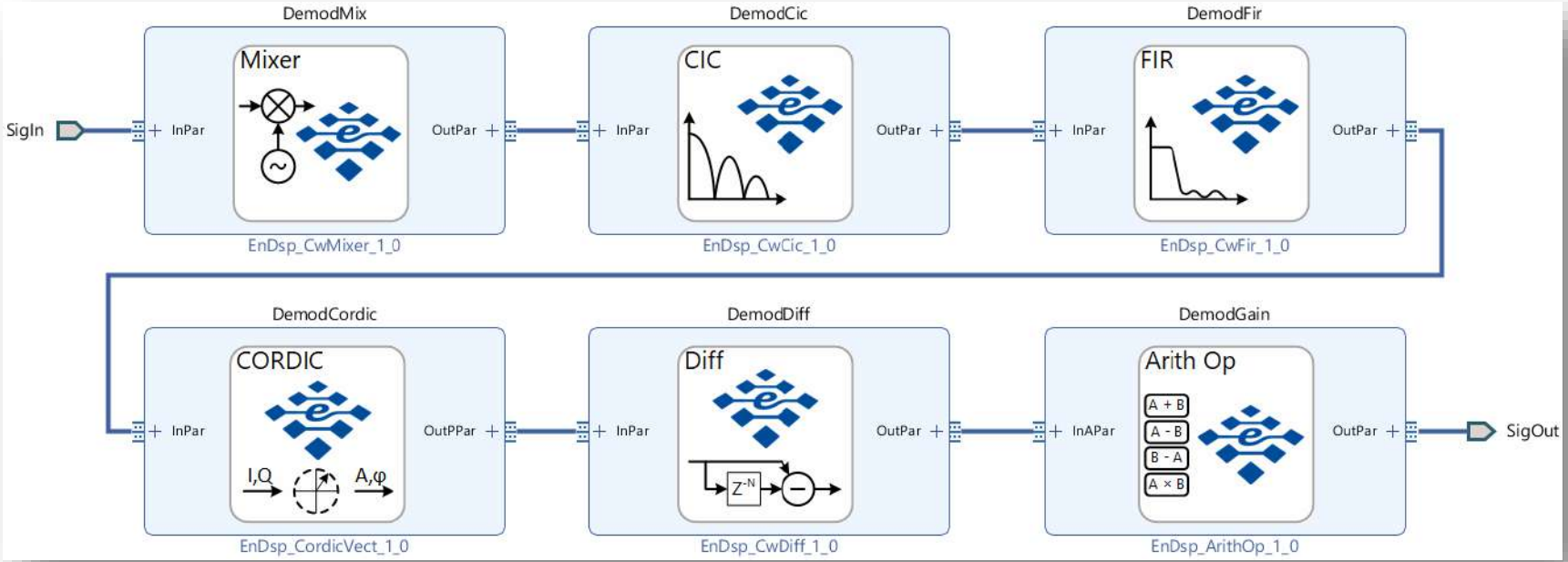
Exactly the same mathematical function requires a totally different implementation at a different sample rate.

Enclustra Universal DSP Library

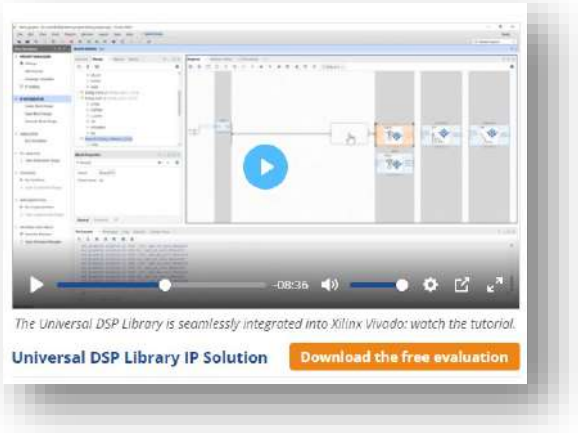
- *Enclustra's Universal DSP Library IP:*
  - *FIR filters*
  - *CIC filters*
  - *Mixers*
  - *CORDICs*
  - *Uniform/Gaussian/arbitrary noise generators*
  - *Function approximations ($\sqrt{x}$, $1/x$)*

- *Pipelined and resource-sharing implementations*
  - *TDM and parallel channel handling*

- *Every IP block has a Xilinx IPI wrapper*



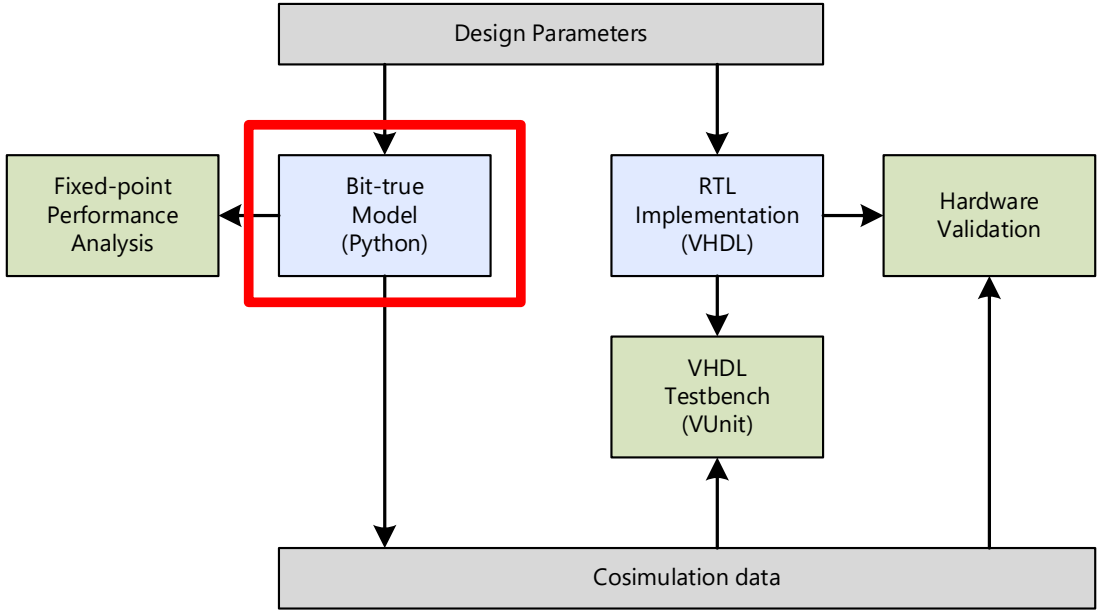
- *More details on our website:*
  - *Demo video (based on Embedded Computing Conference 2021 presentation)*
  - ***Free** evaluation version (encrypted, time-limited)*
  - ***Free** bit-true Python models + full documentation*
  - ***Free** open-source fixed-point library (en_cl_fix)*
  - ***Free** open-source low-level DSP components (psi_fix)*



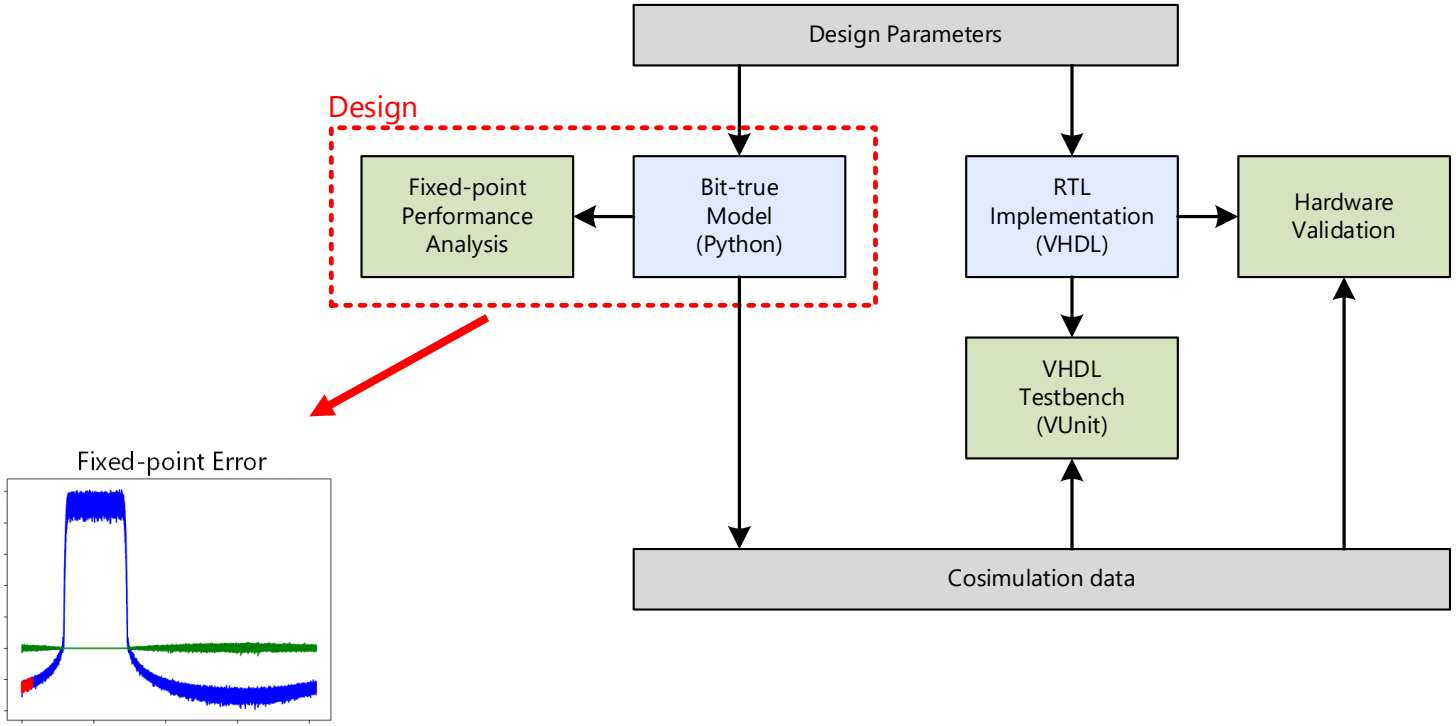
https://www.enclustra.com/en/products/ip-cores/universal-dsp-library/

Bit-true Development Flow

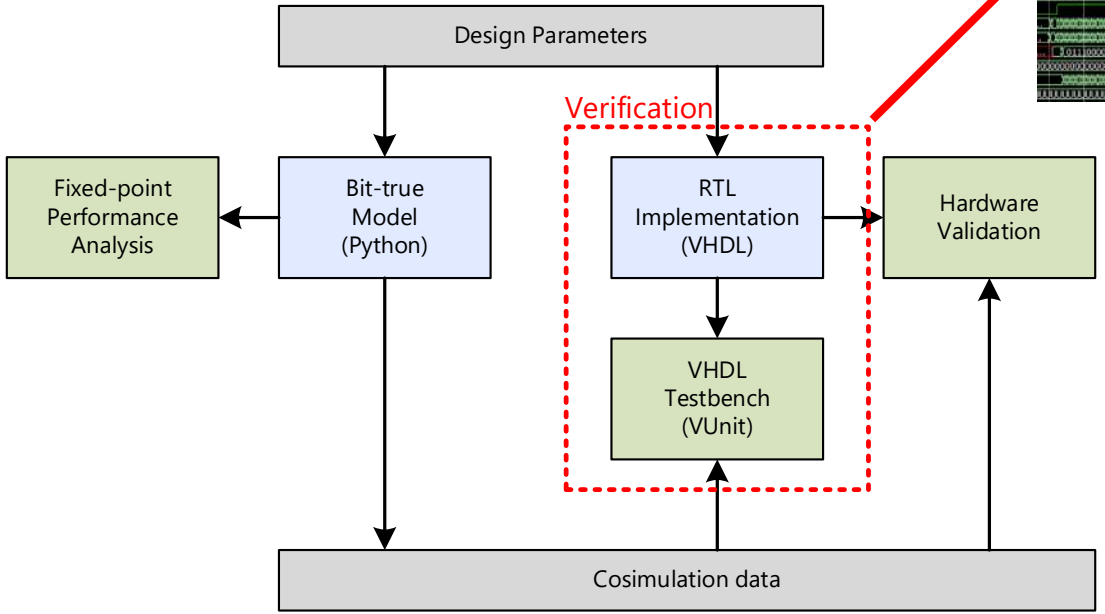
- *Every DSP component has a bit-true software model*



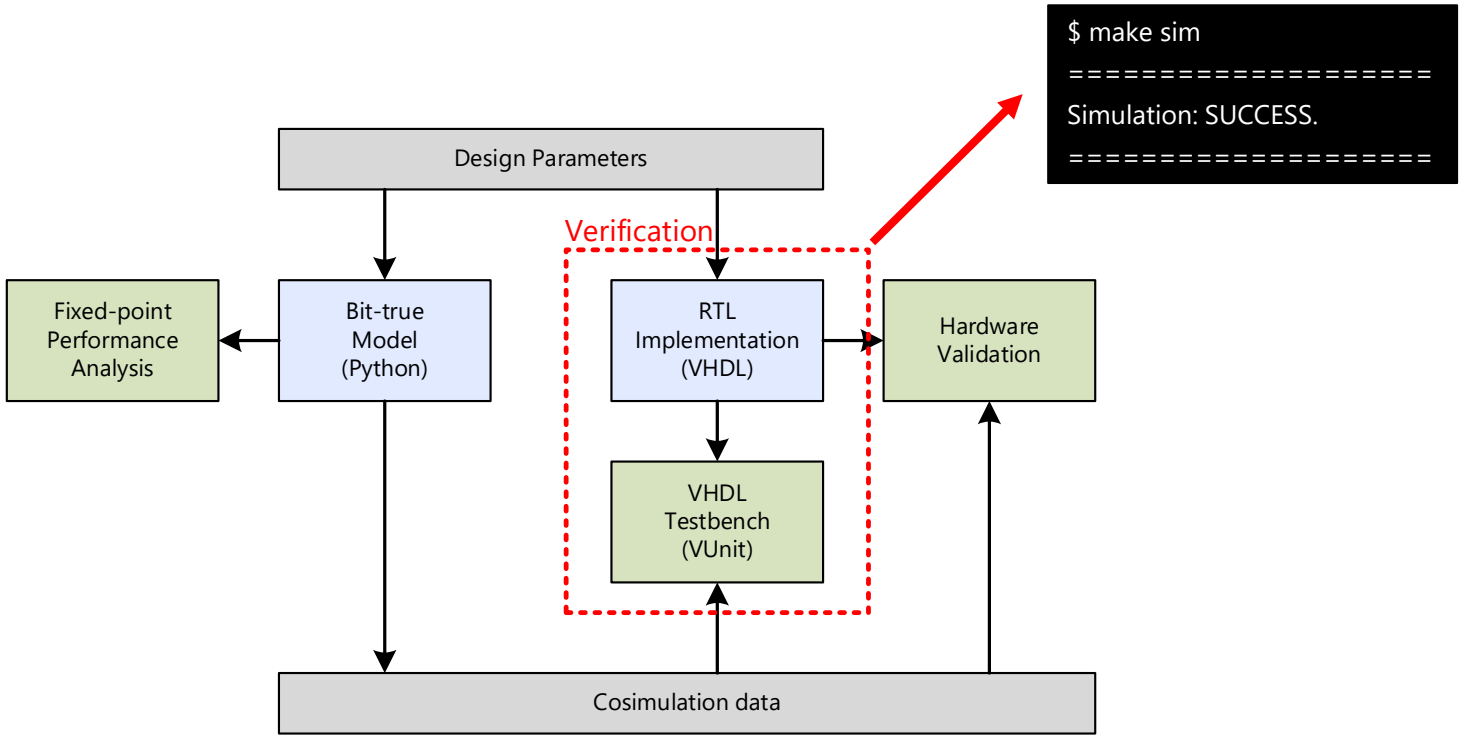
- *Step 1: Design*



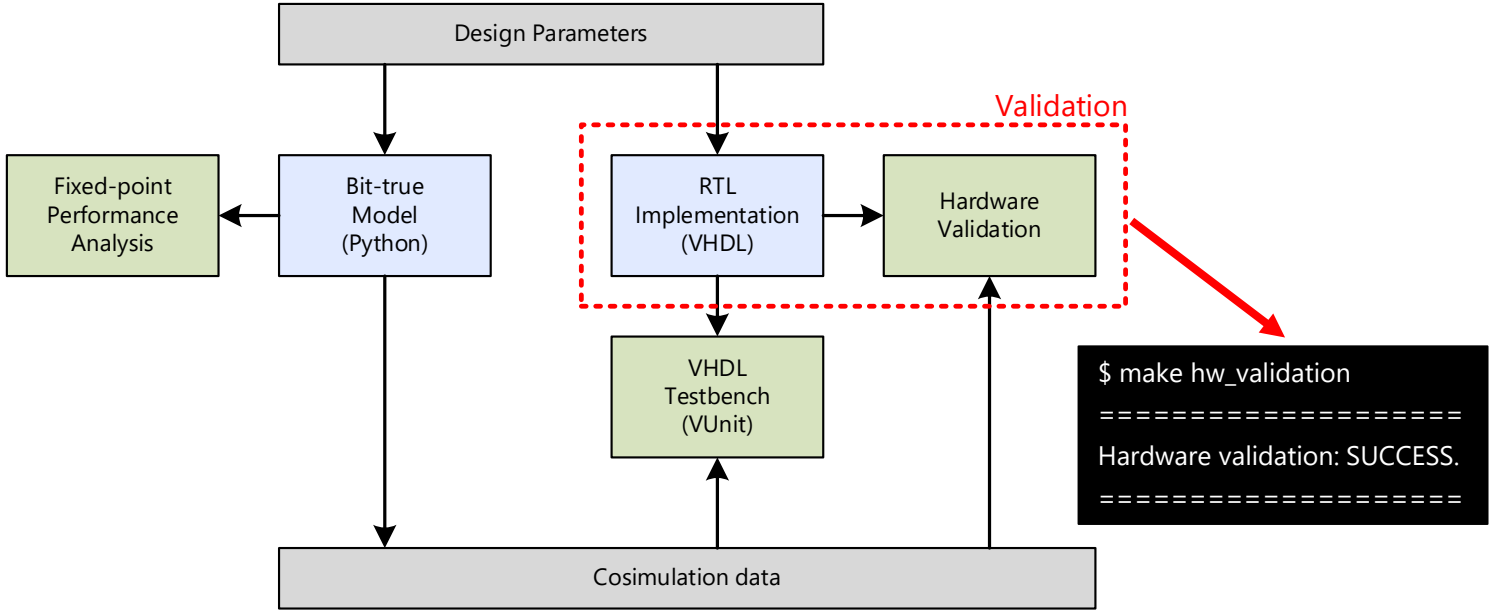
- *Step 2: Verification in simulation*



- *Step 2: Verification in simulation*



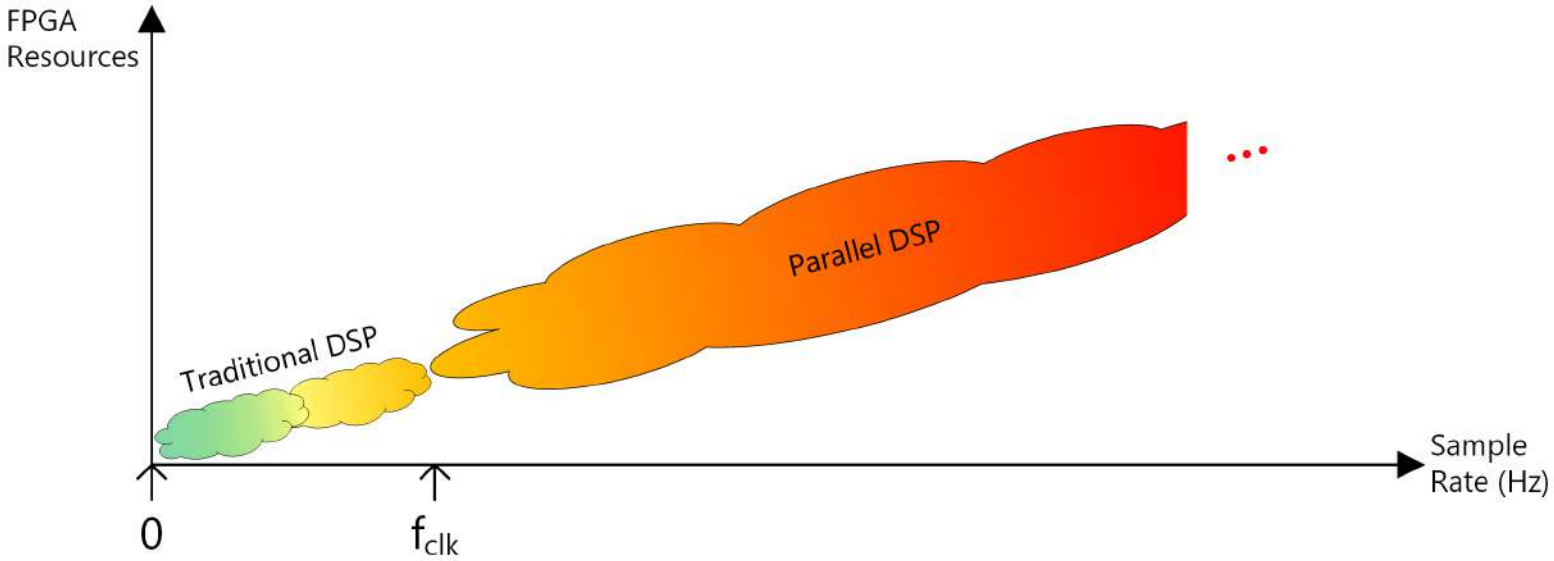
- *Step 3: Validation in hardware*



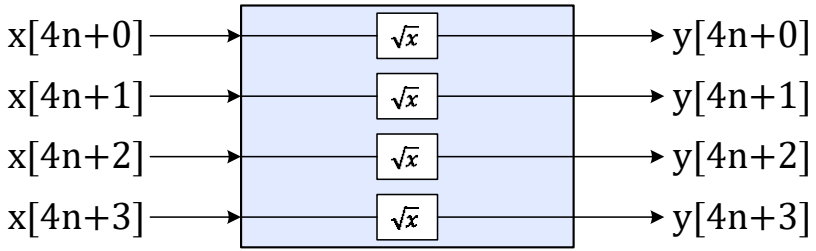
ENCLUSTRA
FPGA SOLUTIONS

Parallel DSP

- *What if sample rate > clock frequency?*
  - *RF sampling ADCs (e.g. Xilinx RFSoC) run at many GS/s*
  - *Traditional DSP techniques may not cover this*

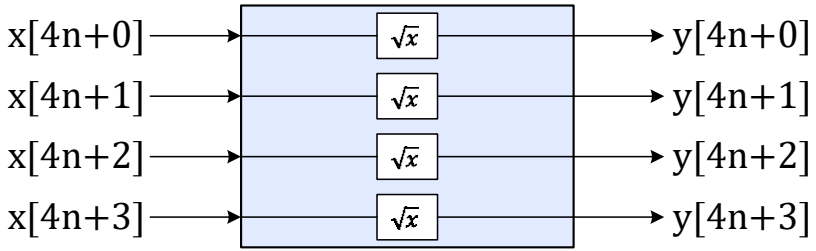


- *Memoryless (per-sample) functions ⇒ no parallelization required*
  - *Basic arithmetic (+, -, ×, ÷, $\sqrt{x}$, $\log(x)$)*
  - *CORDIC (rotate, $\arctan(x)$, $\sqrt{x}$, $\sin(x)$, $\cos(x)$, …)*



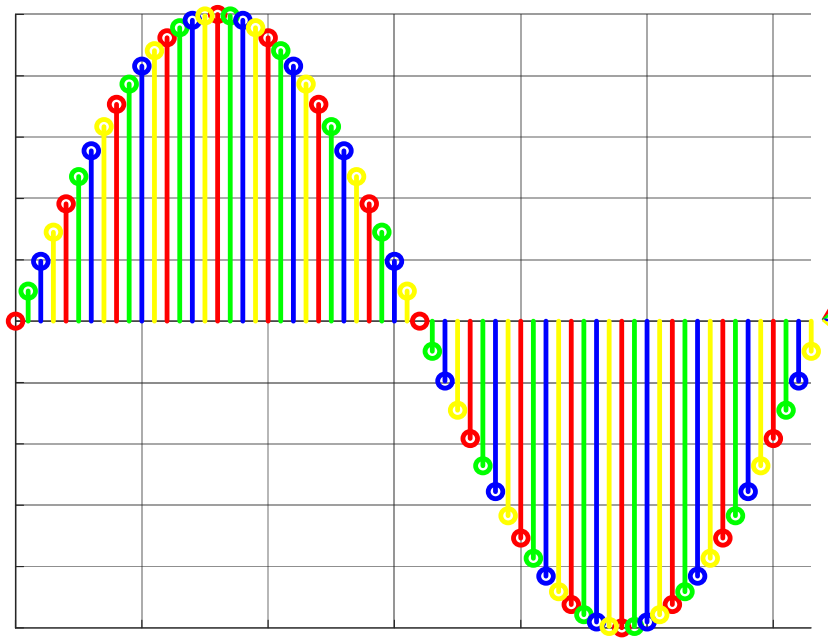
- *Functions with memory ⇒ algorithm must be parallelized*
  - *Pseudo-random number generators*
    - *Algorithm-dependent*
  - *Numerically controlled oscillators (NCOs)*
  - *FIR filters*
  - *FFTs*

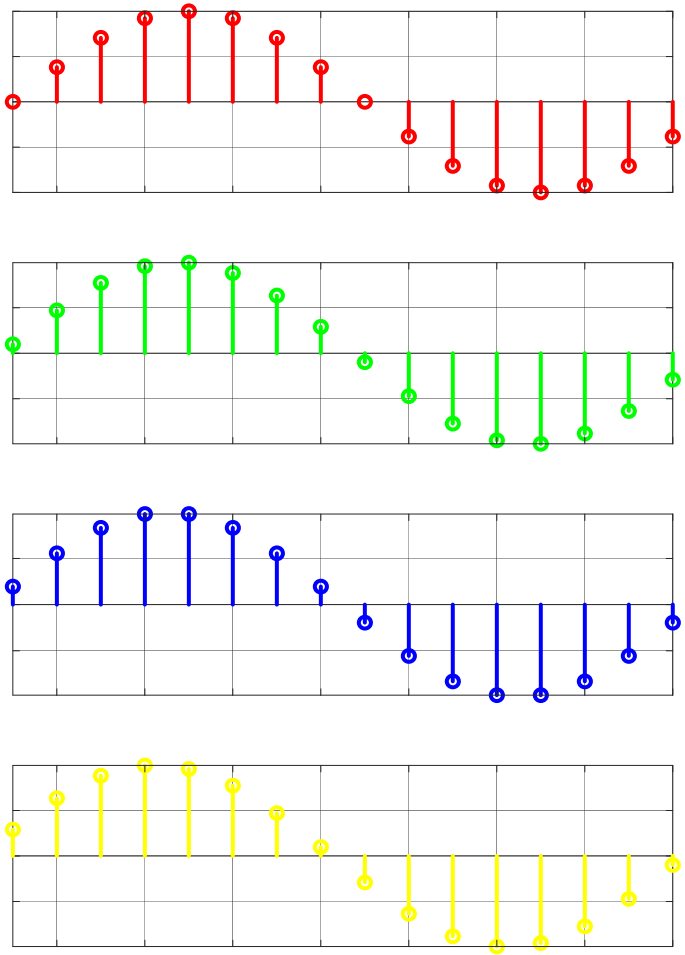
- *Memoryless (per-sample) functions ⇒ no parallelization required*
  - *Basic arithmetic (+, -, ×, ÷, $\sqrt{x}$, $\log(x)$)*
  - *CORDIC (rotate, $\arctan(x)$, $\sqrt{x}$, $\sin(x)$, $\cos(x)$, …)*



- *Functions with memory ⇒ algorithm must be parallelized*
  - *Pseudo-random number generators*
    - *Algorithm-dependent*
  - *Numerically controlled oscillators (NCOs)*
  - *FIR filters*
  - *FFTs*

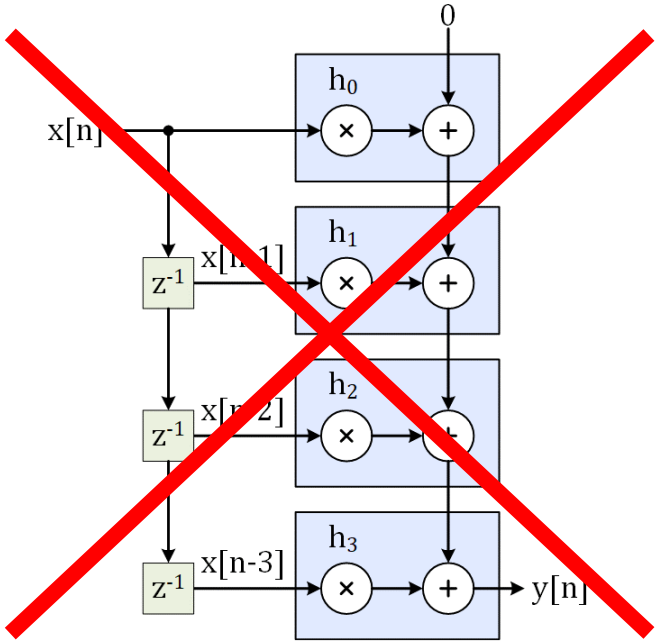
- ***Example 1**: 4-Parallel Sinewave Generator*
  - *Use 4 pipelined sinewave generators in parallel*
  - *Set initial phases and frequencies appropriately*

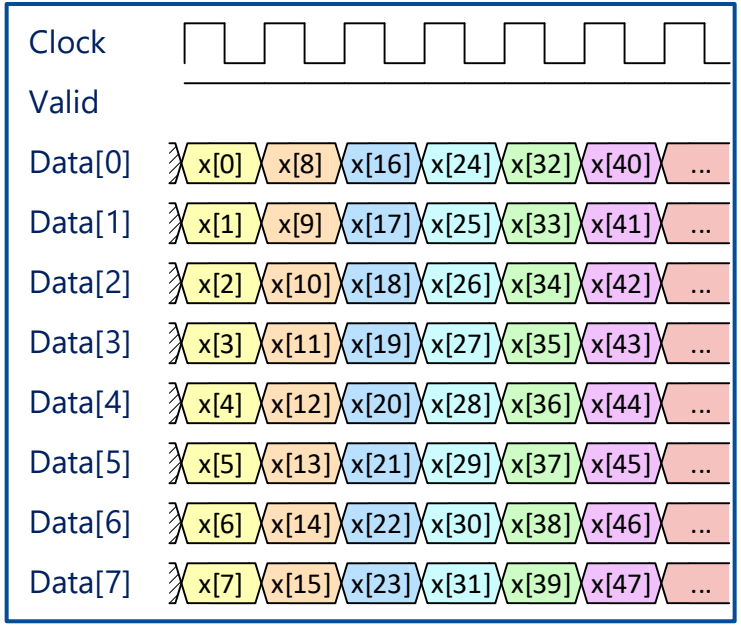




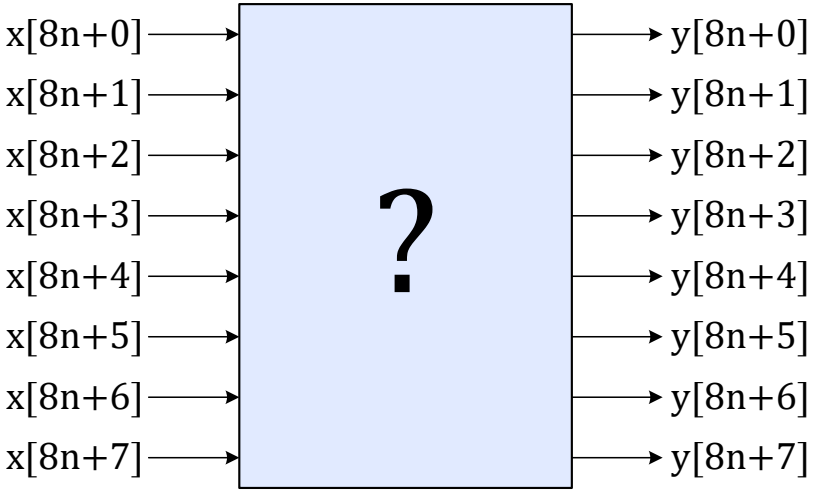
Parallel DSP Example 2

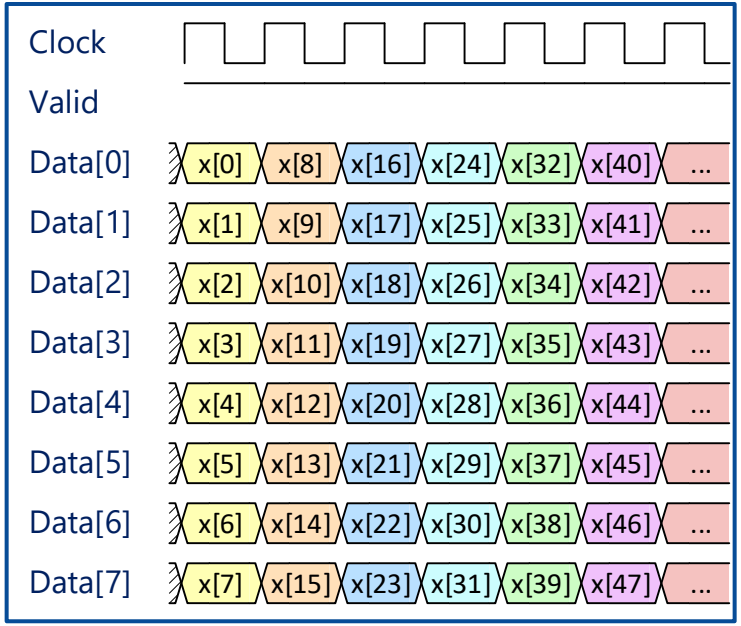
- ***Example 2**: 8-Parallel FIR Filter*
  - *Ready for 8 valid inputs every clock cycle*
  - *How should we arrange the 32 multipliers?*





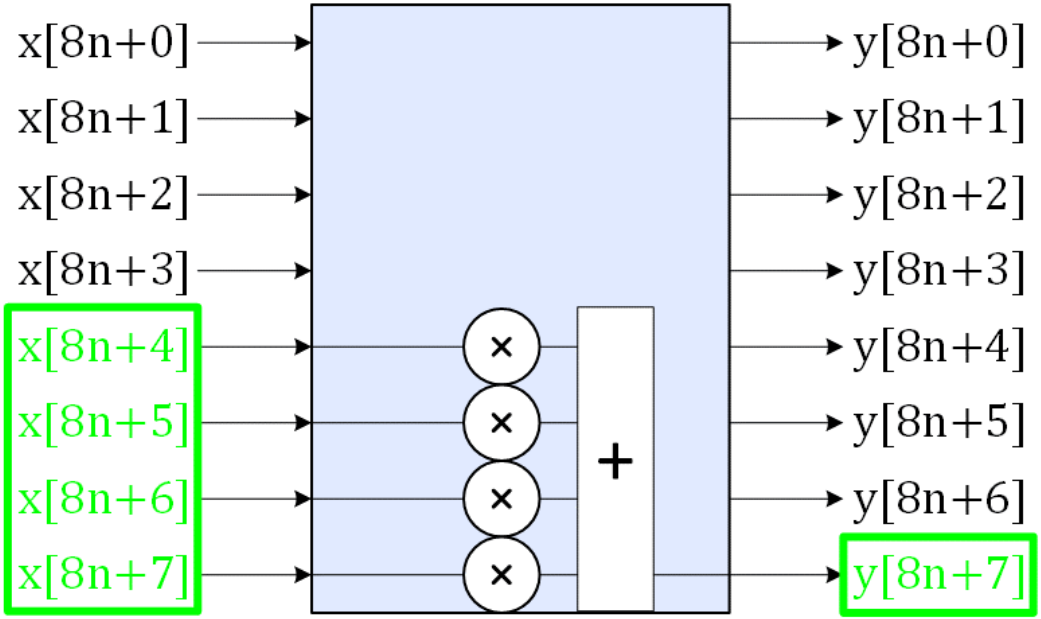
- ***Example 2**: 8-Parallel FIR Filter*
  - *Ready for 8 valid inputs every clock cycle*
  - *How should we arrange the 32 multipliers?*



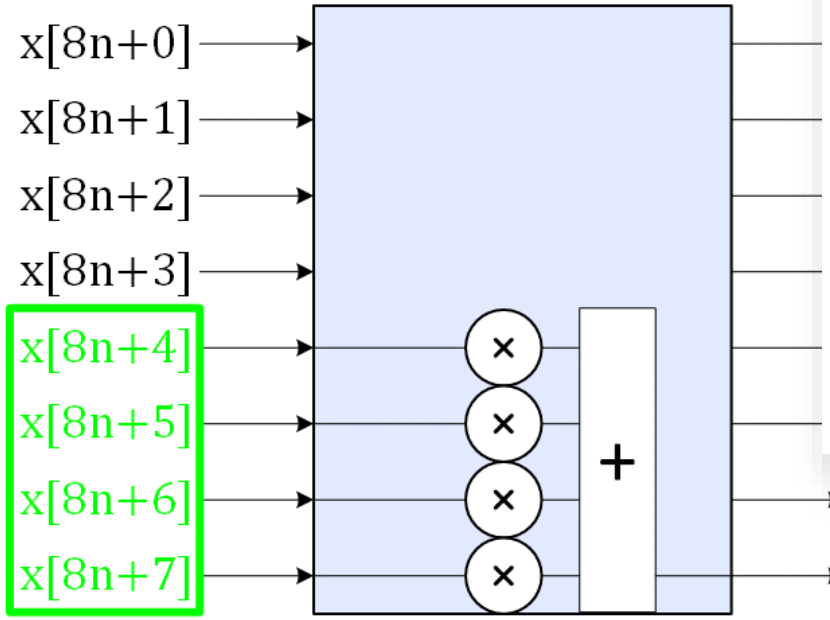


8-Parallel FIR Filter

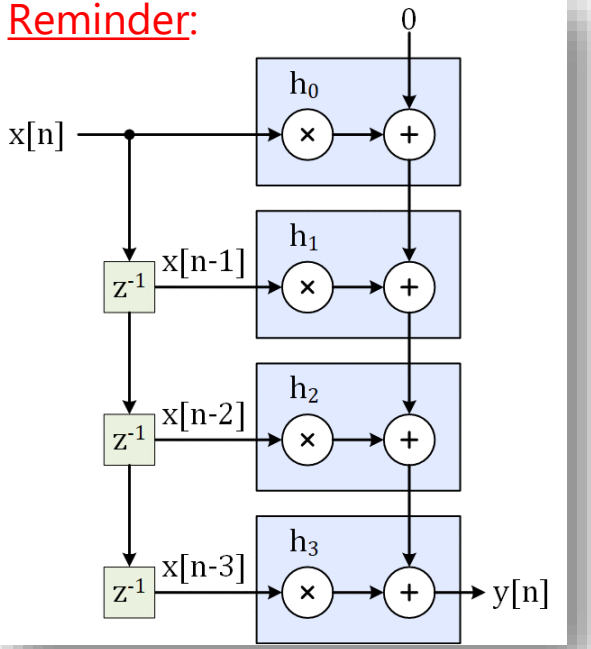
- *Start with the last output:*



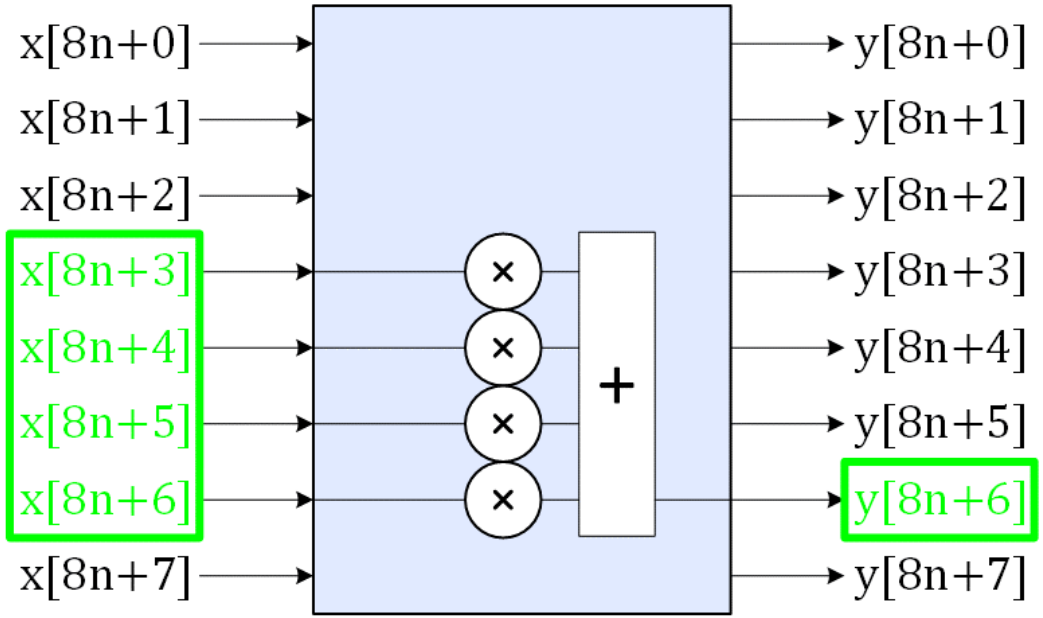
- *Start with the last output:*



Reminder:

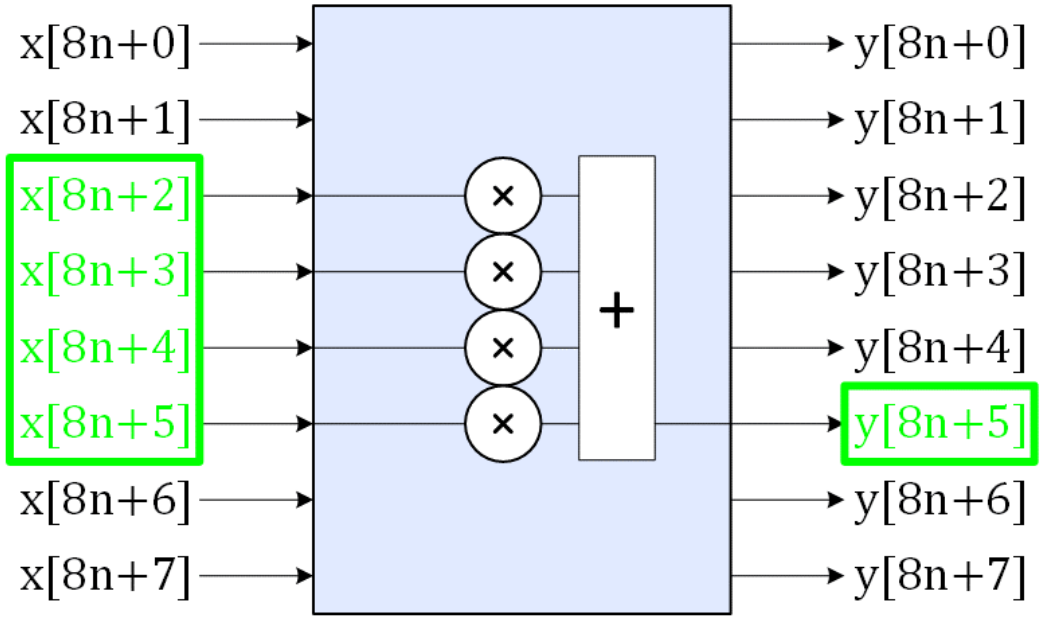


- *Similarly:*


x[8n+0]
x[8n+1]
x[8n+2]
x[8n+3]
x[8n+4]
x[8n+5]
x[8n+6]
x[8n+7]
×
×
×
×
+
y[8n+0]
y[8n+1]
y[8n+2]
y[8n+3]
y[8n+4]
y[8n+5]
y[8n+6]
y[8n+7]

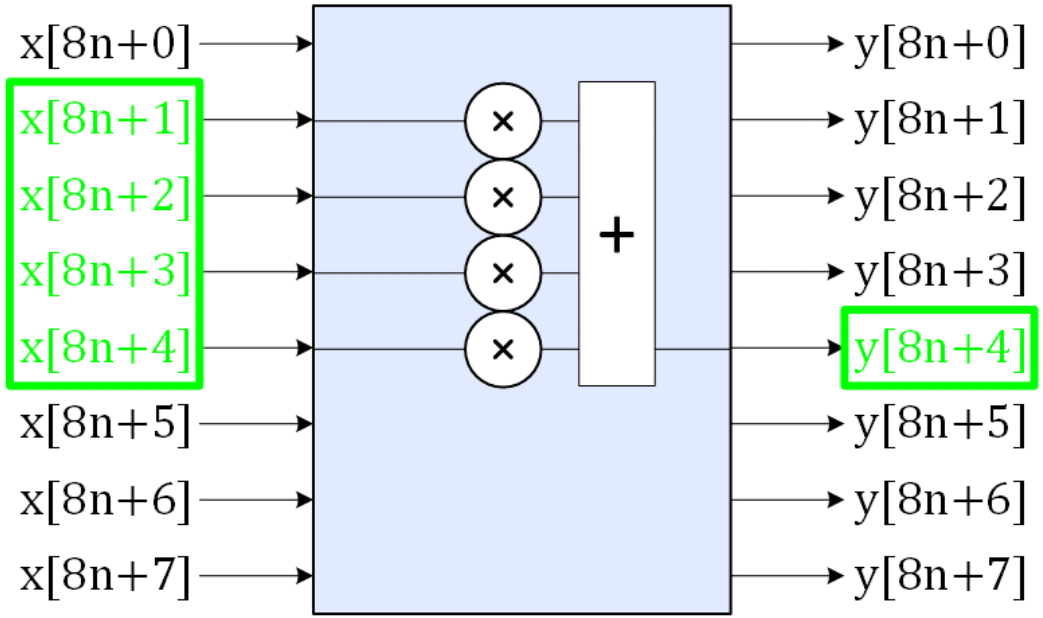
8-Parallel FIR Filter

- *Similarly:*



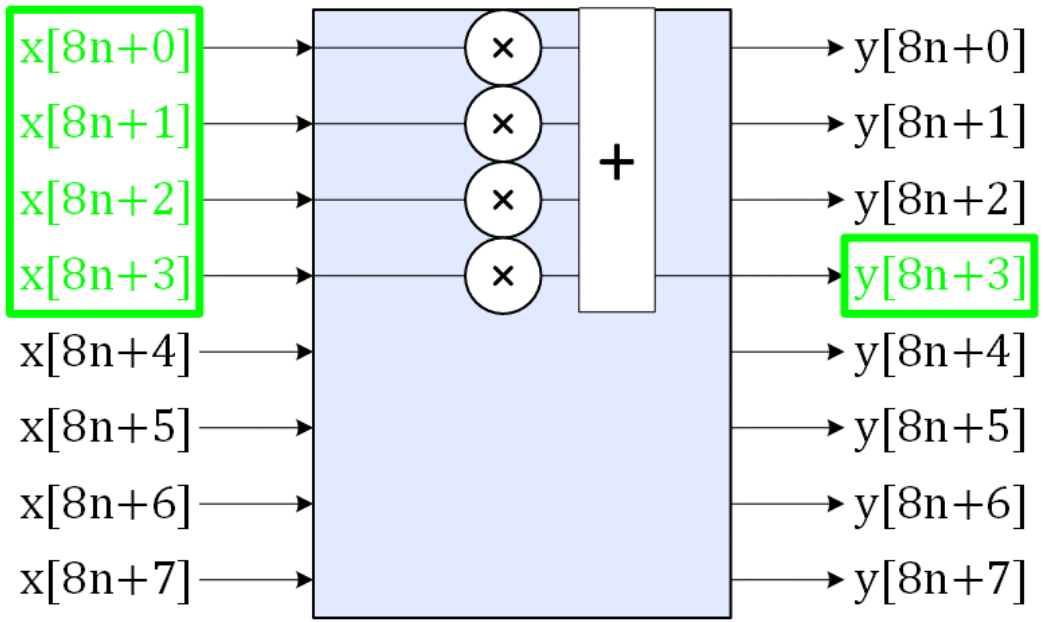
8-Parallel FIR Filter

- *Similarly:*

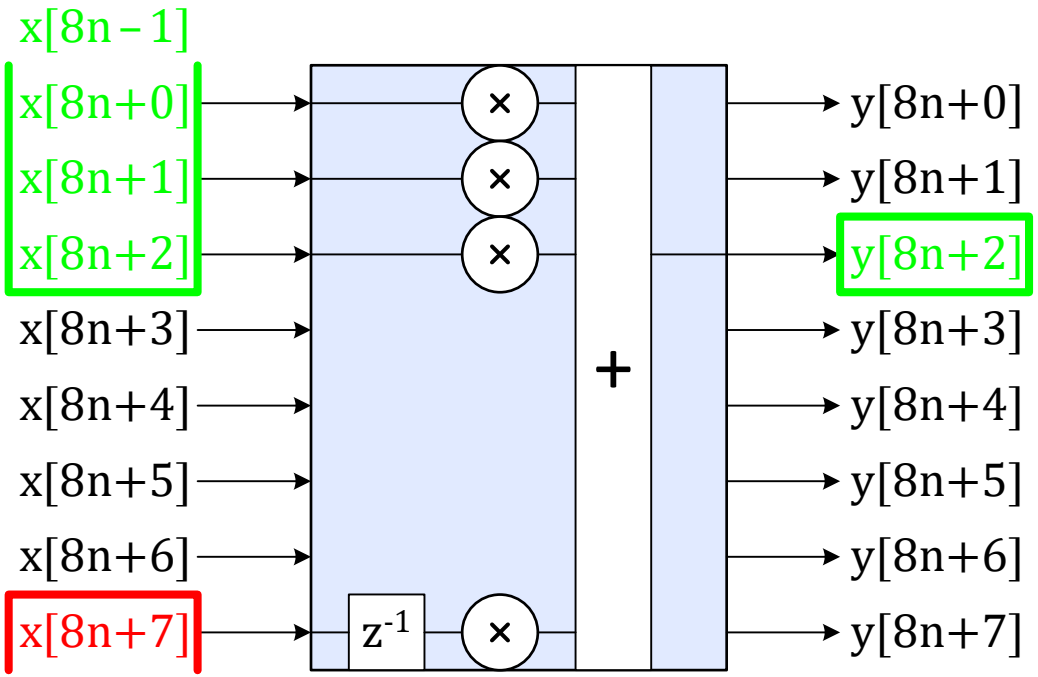


8-Parallel FIR Filter

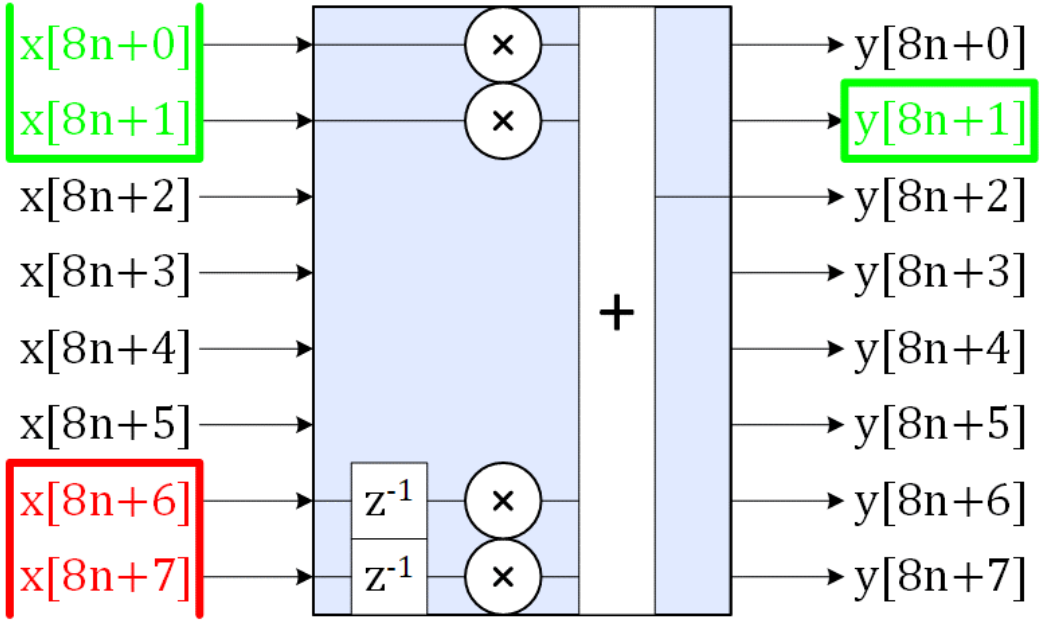
- *Similarly:*



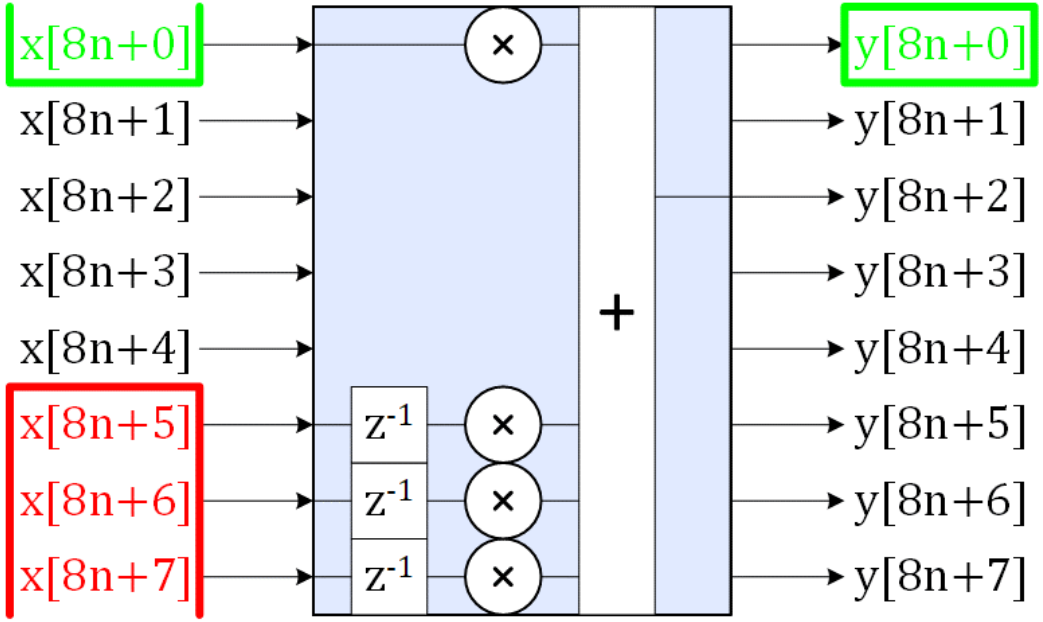
- *Be careful when wrapping to the previous data beat:*



- *Similarly:*

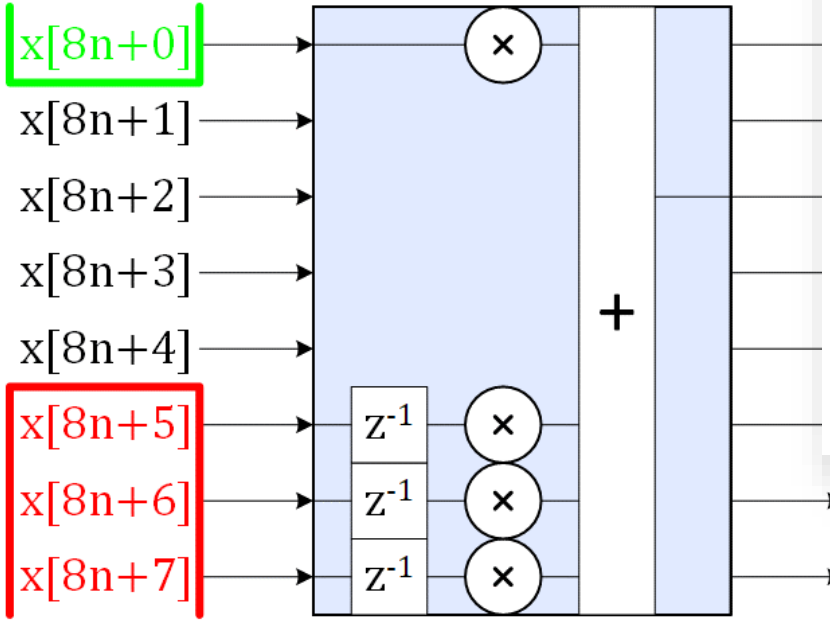

x[8n+0]
x[8n+1]
x[8n+2]
x[8n+3]
x[8n+4]
x[8n+5]
x[8n+6]
x[8n+7]
z-1
z-1
×
×
×
×
+
y[8n+0]
y[8n+1]
y[8n+2]
y[8n+3]
y[8n+4]
y[8n+5]
y[8n+6]
y[8n+7]

- *Similarly:*

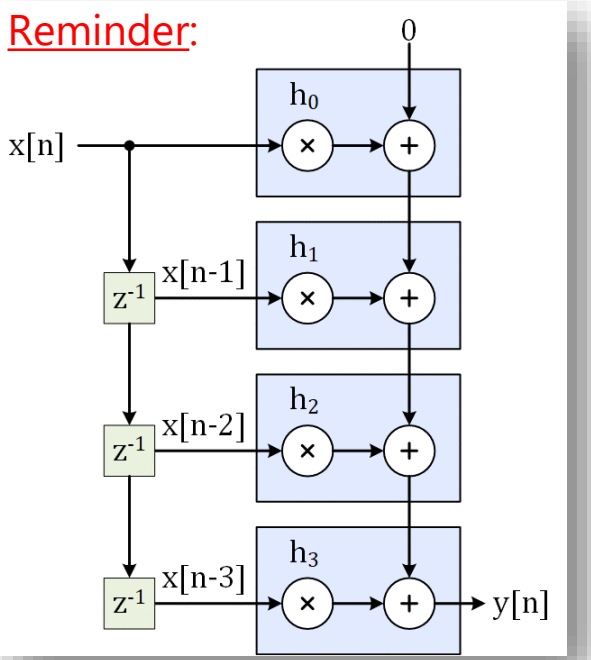

x[8n+0]
x[8n+1]
x[8n+2]
x[8n+3]
x[8n+4]
x[8n+5]
x[8n+6]
x[8n+7]
z-1
z-1
z-1
×
×
×
×
+
y[8n+0]
y[8n+1]
y[8n+2]
y[8n+3]
y[8n+4]
y[8n+5]
y[8n+6]
y[8n+7]

8-Parallel FIR Filter

- *Similarly:*

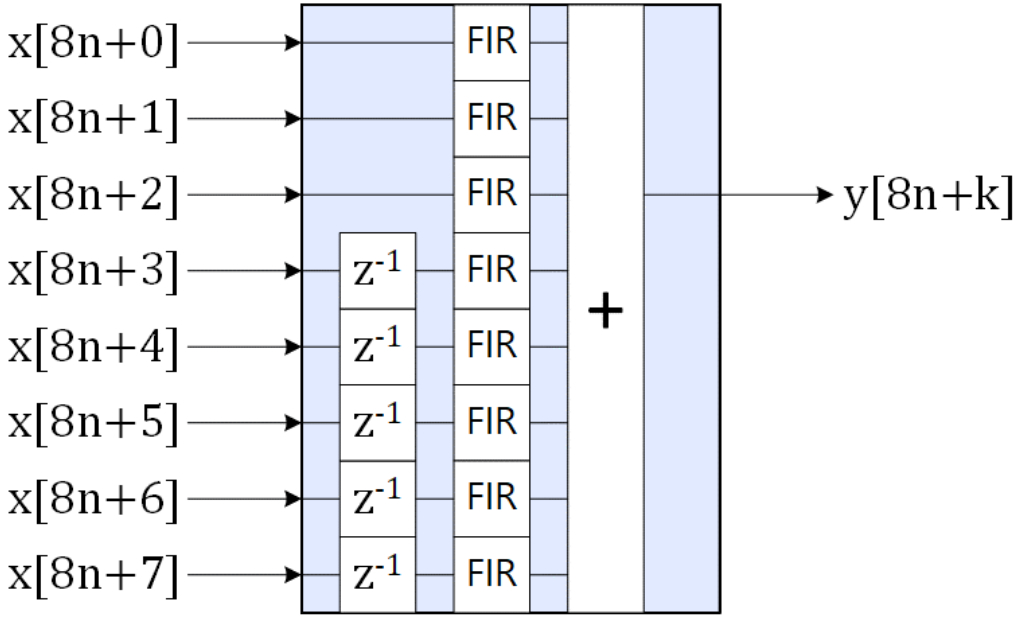


Reminder:



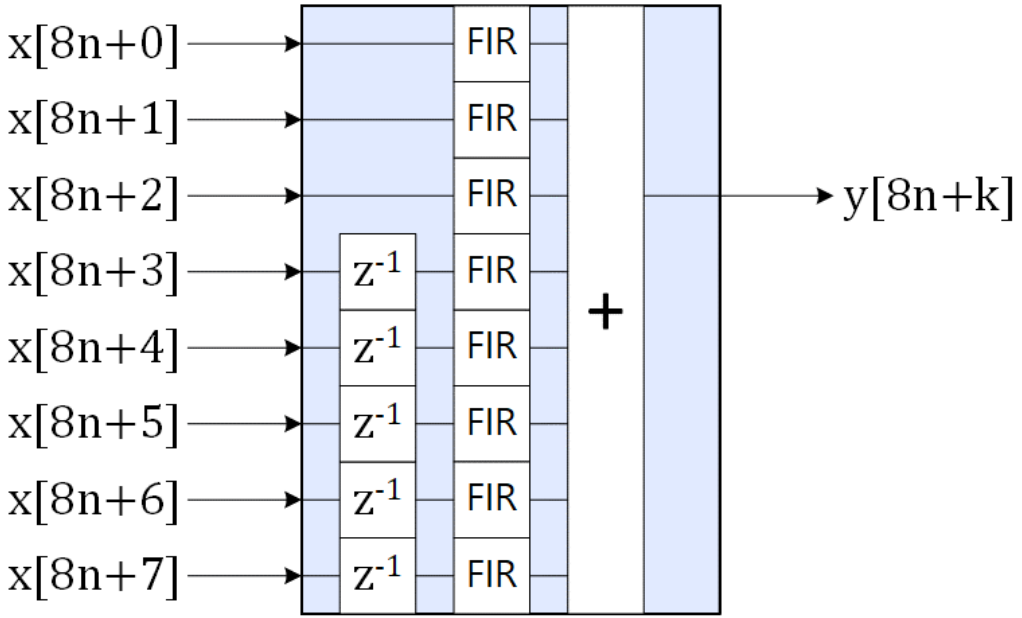
8-Parallel FIR Filter

- *In general, the filter length (N) may be longer than the parallelization factor (P)*
- *Therefore, we need $P^2 = 64$ pipelined FIR subfilters to implement one 8-parallel filter*
- *The average subfilter length is N/P*



8-Parallel FIR Filter

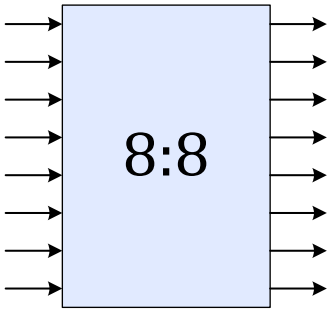
- *In general, the filter length (N) may be longer than the parallelization factor (P)*
- *Therefore, we need $P^2 = 64$ pipelined FIR subfilters to implement one 8-parallel filter*
- *The average subfilter length is N/P*

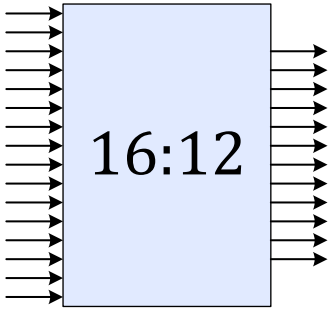


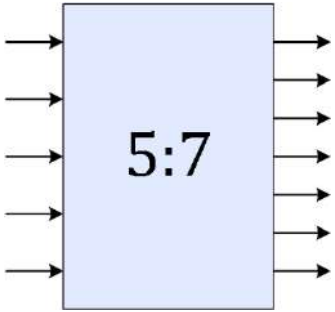
Exactly the same mathematical function requires a totally different implementation at a different sample rate.

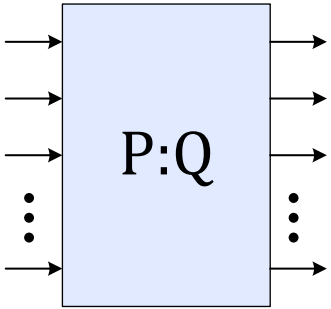
Parallel Polyphase FIR Filters

- *With minor modifications, this result generalizes to all parallel polyphase resampling filters (P inputs → Q outputs)*
- *A large family of parallel filters can be built* using a traditional FIR filter (many instances) and relatively little extra logic*



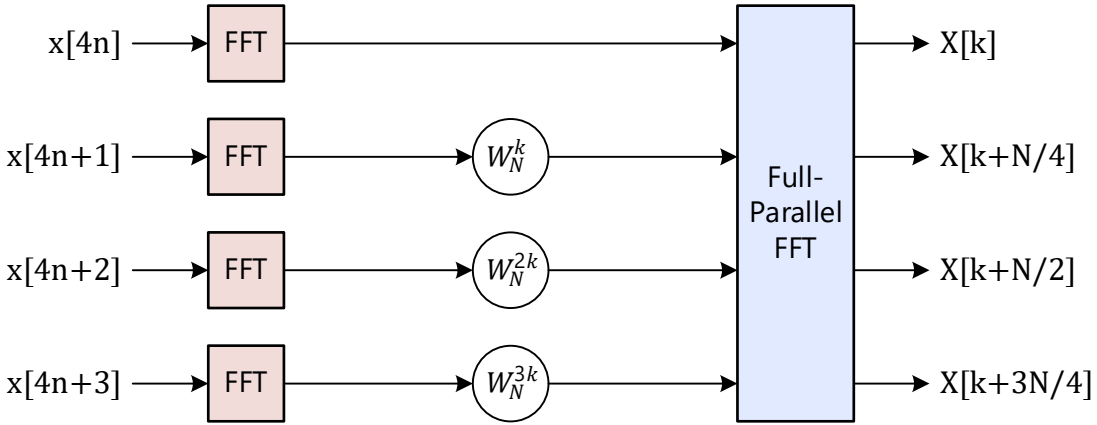




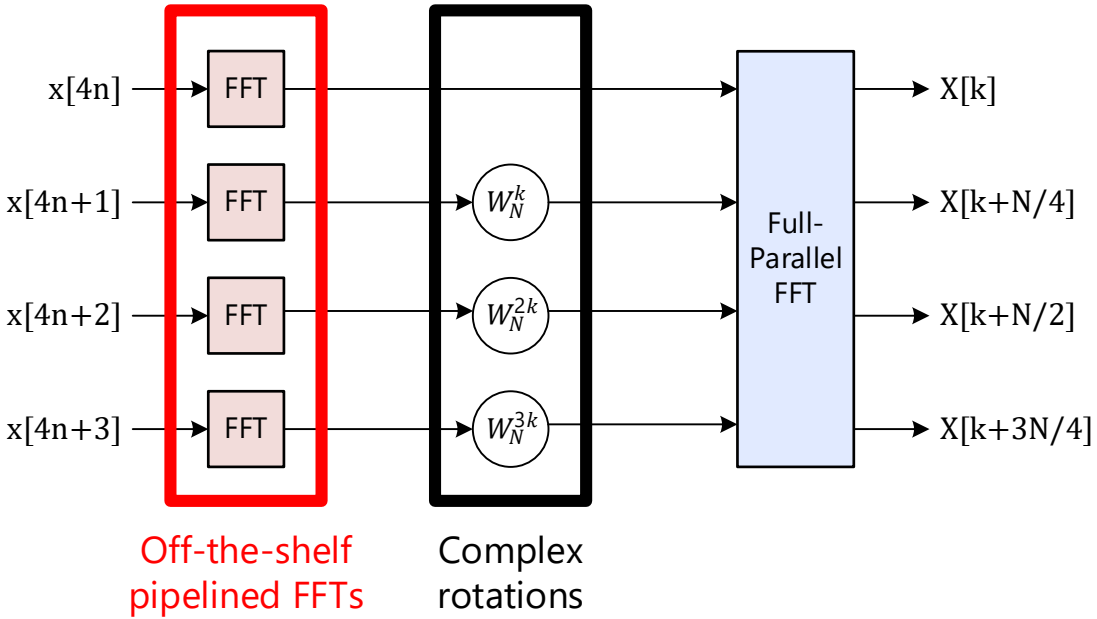


**Efficient optimization of subfilter symmetry/anti-symmetry needs care.*

- ***Example 3**: 4-Parallel Fast Fourier Transform (FFT)*
  - *The parallel FFT has been studied extensively in the academic literature*
  - *A very useful decomposition for FPGA engineers:*



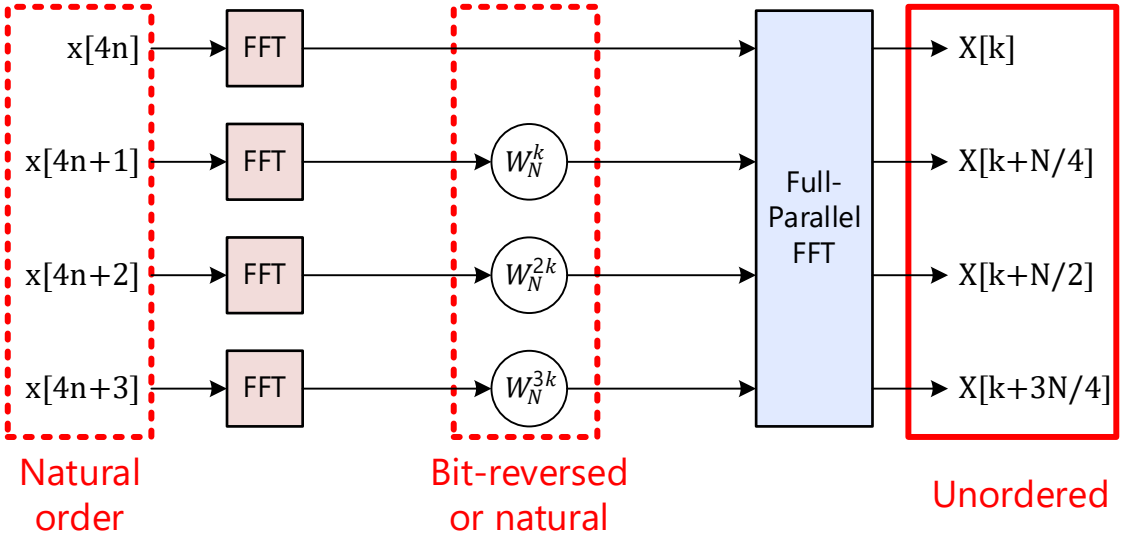
- ***Example 3**: 4-Parallel Fast Fourier Transform (FFT)*
  - *The parallel FFT has been studied extensively in the academic literature*
  - *A very useful decomposition for FPGA engineers*:*



**Huge resource savings are possible with a hand-written pipelined FFT.*

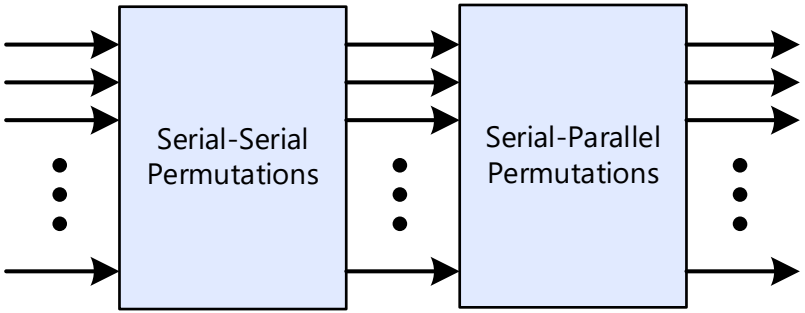
Parallel FFT Reorder Buffer

- *FFT algorithms typically output the data in the "wrong" (bit-reversed) order*
- *Reordering the sub-FFT outputs to natural order doesn't fix the parallel output order*
- *A specific parallel reordering algorithm is needed*



Parallel FFT Reorder Buffer

- *Parallel FFT reorder buffers have been studied extensively in the academic literature*
- *[1] and [2] describe a very simple and efficient implementation*

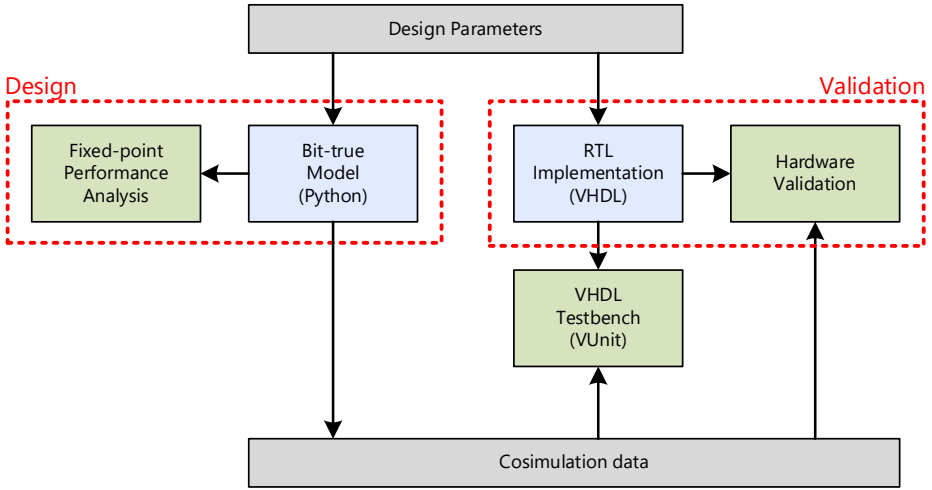


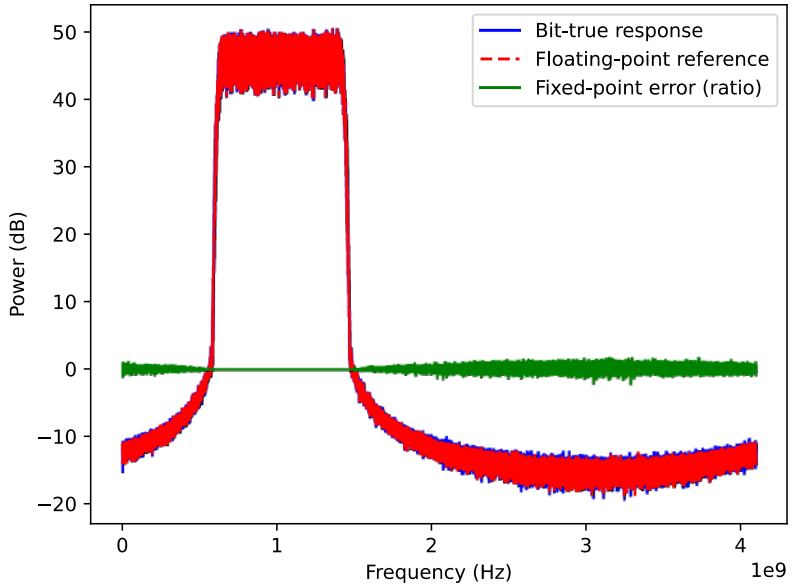
[1] "Continuous-flow Parallel Bit-Reversal Circuit for MDF and MDC FFT Architectures", IEEE Trans. Circuits and Systems, Cheng et al., 2014.

[2] "Multiplexer and Memory-Efficient Circuits for Parallel Bit Reversal", IEEE Trans. Circuits and Systems, Garrido, 2019.

Parallel FFT Bit-True Development

- *A 128k-point 8-parallel FFT may be too slow to simulate in VHDL*
  - *Up to ~1 day of simulation time per FFT frame*
- *The bit-true software model solves this issue*





Summary

- *Reviewed traditional DSP design challenges*
  - *Enclustra's Universal DSP Library IP*
- *Investigated parallel DSP design challenges*
  - *Example 1: Parallel NCO (can be built using many pipelined NCOs)*
  - *Example 2: Parallel FIR filter (can be built using many pipelined FIR filters + extra logic)*
  - *Example 3: Parallel FFT (can be built using many pipelined FFTs + extra logic)*

Everything FPGA.